BatLab Basic Project Kit – Potentiometer – Blink an LED (analogRead)

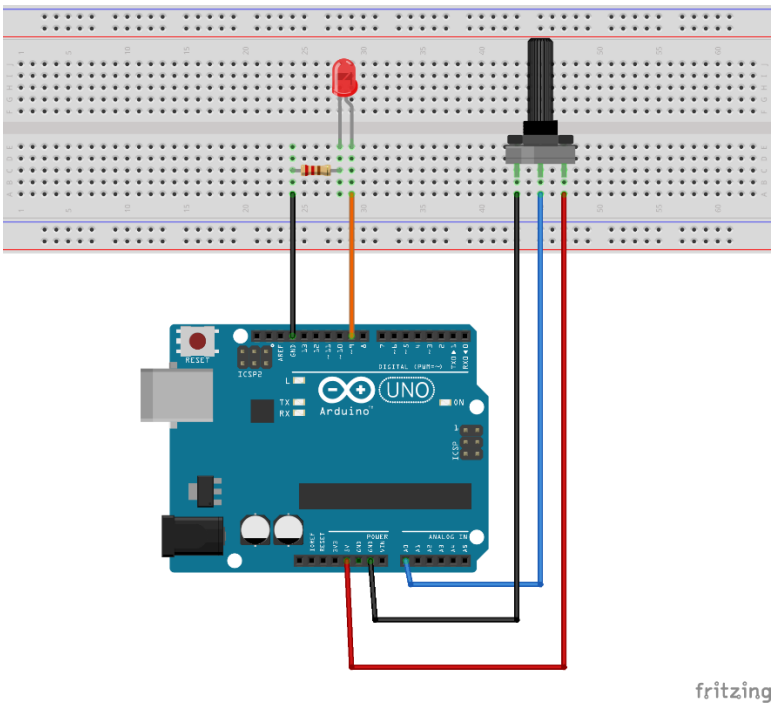
HOW IT WORKS

A potentiometer, or "pot" for short, is a control knob. It's the same type of control you'd use to change volume, dim a lamp, etc. A potentiometer changes resistance as it is turned. By using it as a "voltage divider," the Arduino can sense the position of the knob and use that value to control whatever you wish (like the blink rate of an LED, as we're doing here).

PARTS

- Arduino Uno, breadboard, and jumper wires
- Potentiometer (knob)
- LED (any color)
- 220Ω resistor (red marking)

CIRCUIT



CODE

```
/* POTENTIOMETER - BLINK AN LED

Hardware connections:

  Potentiometer:

    Potentiometers have three pins. When we're using it as a
    voltage divider, we connect the outside pins to power and
    ground. The middle pin will be the signal (a voltage which
    varies from 0 Volts to 5 Volts depending on the position of
    the knob).

    Connect the middle pin to ANALOG IN pin 0 on the Arduino.
    Connect one of the outside pins to 5V.
    Connect the other outside pin to GND.

    (TIP: if once your program is running, the knob feels
    "backwards", you can swap the 5V and GND pins to reverse
    the direction.)

  LED:

  Connect the positive side of your LED (longer leg) to
   Arduino DIGITAL pin 9

  Connect the negative side of your LED (shorter leg) to a 220 Ohm
resistor

  Connect the other side of the resistor to ground.

This sketch is a modification of Circuit #2 from the
Sparkfun Inventor's Kit, written by Sparkfun Electronics.
http://learn.sparkfun.com/

*/


// In this sketch we'll start using "variables."

// A variable is a named number. We'll often use these to store
// numbers that change, such as measurements from the outside
// world, or to make a sketch easier to understand (sometimes a
// descriptive name makes more sense than looking at a number).

// Variables can be different "data types", which is the kind of
// number we're using (can it be negative? Have a decimal point?)
// We'll introduce more data types later, but for the moment we'll
```

```
// stick with good old "integers" (called "int" in your sketch).

// You must "declare" variables before you use them, so that the
// computer knows about them. Here we'll declare two integer
// variables, and at the same time, initialize them to specific
// values. We're doing this so that further down, we can refer to
// the pins by name rather than number.

// Note that variable names are case-sensitive! If you get an
// "(variable) was not declared in this scope" error, double-check
// that you typed the name correctly.

// Here we're creating a variable called "sensorPin" of type "int"
// and initializing it to have the value "0":

int sensorPin = 0;    // The potentiometer is connected to
                      // analog pin 0

int ledPin = 9;      // The LED is connected to digital pin 9


void setup()
{


  pinMode(ledPin, OUTPUT);

  // The above line is the same as "pinMode(9, OUTPUT);"

  // You might be wondering why we're not also configuring
  // sensorPin as an input. The reason is that this is an
  // "analog in" pin. These pins have the special ability to
  // read varying voltages from sensors like the potentiometer.
  // Since they're always used as inputs, there is no need to
  // specifically configure them.
}


void loop()

{
  // First we'll declare another integer variable
  // to store the value of the potentiometer:

  int sensorValue;

  // The Arduino can read external voltages on the analog input
  // pins using a built-in function called analogRead(). This
  // function takes one input value, the analog pin we're using
  // (sensorPin, which we earlier set to 0). It returns an integer
  // number that ranges from 0 (0 Volts) to 1023 (5 Volts).
  // We're sticking this value into the sensorValue variable:
```

```
  sensorValue = analogRead(sensorPin);

  // Now we'll blink the LED like in the first example, but we'll
  // use the sensorValue variable to change the blink speed
  // (the smaller the number, the faster it will blink).

  // Note that we're using the ledPin variable here as well:

  digitalWrite(ledPin, HIGH);     // Turn the LED on

  delay(sensorValue);             // Pause for sensorValue
                                  // milliseconds

  digitalWrite(ledPin, LOW);      // Turn the LED off

  delay(sensorValue);             // Pause for sensorValue
                                  // milliseconds

  // Remember that loop() repeats forever, so we'll do all this
  // again and again.
}
```